

## REMARKS

Reconsideration of this application as amended is respectfully requested.

In the Office Action dated Jan. 4, 2010, claims 1-29 are pending. Claims 1-29 stand rejected. In this response, Claims 1, 8, 15, 17, 20, 23, 26, and 29 have been amended. No new claims have been added. Claim 7 has been canceled. Thus, claims 1-6 and 8-29 remain pending. Support for the amendments can be found throughout the specifications as filed. No new matter has been added.

### **Rejections under 35 U.S.C. § 102(b)**

#### ***Claims 8-9***

Claims 8-9 stand rejected under 35 U.S.C. §102(b) as being anticipated by "Tolerating Memory Latency through Software-Controlled Pre-Execution in Simultaneous Multithreading Processors" by Luk (Luk I). However, applicants respectfully submit that applicants' claims 8-9, as amended, are not anticipated by the cited reference.

Specifically, for example, independent claim 8 includes the limitations:

"wherein the generated software codes include synchronization codes for the one or more helper threads to synchronize with the main thread during the execution, wherein synchronization codes are generated based on a synchronization period for the one or more helper threads to synchronize with each other during the execution, and wherein the synchronization period is based on a distance between the one or more helper threads and the main thread." (emphasis added)

Applicants submit that Luk I fails to teach or suggest the limitation of *wherein the generated software codes include synchronization codes for the one or more helper threads to synchronize with the main thread during the execution, wherein synchronization codes are generated based on a synchronization period for the one or more helper threads to synchronize with each other during the execution, and wherein the synchronization period is based on a distance between the one or more helper threads and the main thread.*

Rather, Luk teaches extending an instruction set with three new instructions to allow a main thread to spawn and terminate a pre-execution thread (Luk, sec. 3.1, page 44). Luk specifically states that only a main thread, i.e. not a pre-execution thread, is allowed to spawn and terminate a pre-execution thread (Luk, sec 3.1, page 44). Luk also describes hardware based heuristics for a hardware to terminate a pre-execution thread. (Luk, sec 3.5, page 46). Thus,

Luk's pre-execution threads may be controlled by a main thread or a hardware mechanism. However, Luk is completely silent about generating software codes including synchronization codes for a helper thread to synchronize with a main thread during execution. Further, Luk I teach that the pre-execution is to stop (PreExecute\_Stop command) at the end of each loop, to prevent runaway threads. Luk I also teach an error avoidance mechanism where the thread is terminated if the next PC is out of range. Neither of these techniques are synchronization methods.

Instead, Luk I teach mechanisms to avoid runaway threads and PC overflow, but merely stopping the threads. In contrast, Applicants' claimed invention requires that the synchronization codes are based on a *period* so that the one or more threads may synchronize with each other, and the period is based on a distance between the helper threads and the main thread. Luk I do not teach or suggest periods being used to aid synchronization.

In order to anticipate a claim, each and every limitation of the claim must be taught by the cited reference. It is respectfully submitted that Luk I fails to disclose the limitations set forth above. Therefore, it is respectfully submitted that independent Claim 8 and dependent Claim 9, as amended, are not anticipated by Luk I.

### **Rejections Under 35 U.S.C. 103(a)**

#### ***Claims 1-2 and 15-16***

Claims 1-2 and 15-16 are rejected under 35 U.S.C. 103(a) as being unpatentable over "Tolerating Memory Latency through Software-Controlled Pre-Execution in Simultaneous Multithreading Processors" by Luk (Luk I) in view of US 2003/0084433 to Luk et al. (Luk II). Applicants respectfully traverse this rejection, which should be withdrawn for at least the reasons set forth herein.

As amended, independent Claims 1 and 15 require, at least, *wherein selecting the code region further comprises determining a synchronization period for the helper thread to synchronize the main thread and the helper thread, the helper thread performing its tasks within the synchronization period*. As discussed above, Luk I do not teach or suggest this type of synchronization among threads. Further, Luk II fails to teach this type of synchronization.

### ***Claims 3-4***

Claims 3-4 are rejected under 35 U.S.C. § 103(a) as being unpatentable over "Tolerating Memory Latency through Software-Controlled Pre-Execution in Simultaneous Multithreading Processors" by Luk (Luk I) in view of US 2003/0084433 to Luk et al. (Luk II) in view of "Exploiting Hardware Performance Counters with Flow and Context Sensitive Profiling" by Ammons et al. (Ammons). Applicants respectfully traverse this rejection, which should be withdrawn for at least the reasons set forth herein.

Claims 3-4 include the limitations of parent Claim 1. As discussed above, neither Luk I nor Luk II teach or suggest the type of synchronization codes as recited in the claims. Ammons teach profiling of code to analyze and predict paths likely to have cache misses. However, Ammons is silent on techniques involving helper threads or synchronization of threads. Thus, the cited references, either alone or in combination, fail to teach the recited limitations.

### ***Claims 5-7, 17-19, 22-25 and 28-29***

Claims 5-7, 17-19, 22-25 and 28-29 are rejected under 35 U.S.C. § 103(a) as being unpatentable over "Tolerating Memory Latency through Software-Controlled Pre-Execution in Simultaneous Multithreading Processors" by Luk (Luk I) in view of US 2003/0084433 to Luk et al. (Luk II) in view of "Data Prefetching by Dependence Graph Precomputation" by Annavaram et al. (Annavaram). Applicants respectfully traverse this rejection, which should be withdrawn for at least the reasons set forth herein.

Regarding Claim 5, Annavaram seem to teach generating dependence graphs as runtime (see at least the *Abstract*, line 15). Dependencies are determined based on instructions in the instruction fetch queue (see, at least, page 52, second full paragraph). Further, it seems that the dependence graph generator of Annavaram is a *hardware* construct added to the system (See Figure 1, page 53). In contrast, Applicants' claimed invention builds dependent graphs from the source code, not at runtime. Analyzing source code by a compiler is not the same as hardware to generate graphs during runtime, of instructions in the fetch queue. Thus, the cited art fails to teach or suggest all of the limitations of the recited claims.

Independent Claims 17, 23 and 29 are amended to require *wherein the one or more helper threads are synchronized with the main thread based on a synchronization period determined for the one or more helper threads, so that the one or more helper threads perform*

*the one or more computations during the synchronization period.* As discussed above, the cited art fails to teach or suggest a synchronization period determined for the helper threads. Thus, Claims 17, 23, 29 and their progeny are believed allowable.

Further, the Office Action notes that Luk discloses ... determining a synchronization period for the helper thread to synchronize the main thread with the helper thread ... (pg. 46, col. 1, 2<sup>nd</sup> par. A pre-execution thread must be terminated if its next PC is out of the acceptable range" (Office Action, page 10). It appears that the Office Action alleges Luk's acceptable address range is a synchronization period for a thread to synchronize with a main thread within the synchronization period. Applicants respectfully disagree.

Instead, Luk terminates a pre-execution thread if its next PC is out of an acceptable range imposed by an operating system to preserve correctness (Luk, pg. 46, col. 1, 2<sup>nd</sup> par). Here, Luk merely exhibits the common-sense approach that a thread should be terminated if the PC for the next instruction falls outside the acceptable range of executable addresses of the operating system. Whether the PC of a thread falls outside the acceptable address range imposed by an operating system has nothing to do with whether the thread synchronizes with a main thread. Thus, Luk does terminate a thread when the PC of the thread is out of an acceptable range. It is respectfully submitted that one with ordinary skill in the art would not recognize determining a synchronization *period* for a helper thread to synchronize a main thread with the helper thread within the synchronization period based on Luk's teaching.

Furthermore, there are neither motivations nor suggestions to combine the cited references. Therefore, for the reasons similar to those discussed above, these claims are patentable over the cited references.

### ***Claims 10-11***

Claims 10-11 are rejected under 35 U.S.C. § 103(a) as being unpatentable over "Tolerating Memory Latency through Software-Controlled Pre-Execution in Simultaneous Multithreading Processors" by Luk (Luk I) in view of "Exploiting Hardware Performance Counters with Flow and Context Sensitive Profiling" by Ammons et al. (Ammons). Applicants respectfully traverse this rejection, which should be withdrawn for at least the reasons set forth herein, at least as described above, for Claims 3-4.

### ***Claims 12-14***

Claims 12-14 are rejected under 35 U.S.C. 103(a) as being unpatentable over "Tolerating Memory Latency through Software-Controlled Pre-Execution in Simultaneous Multithreading Processors" by Luk (Luk I) in view of "Data Prefetching by Dependence Graph Precomputation" by Annavaram et al. (Annavaram). Applicants respectfully traverse this rejection, which should be withdrawn for at least the reasons set forth herein, at least as described above, for Claims 5-7.

### ***Claims 20-21 and 26-27***

Claims 20-21 and 26-27 are rejected under 35 U.S.C. 103(a) as being unpatentable over "Tolerating Memory Latency through Software-Controlled Pre-Execution in Simultaneous Multithreading Processors" by Luk (Luk I) in view of US 2003/0084433 to Luk et al. (Luk II) in view of "Data Prefetching by Dependence Graph Precomputation" by Annavaram et al. (Annavaram) in view of US 7,243,267 to Klemm et al. (Klemm).

The Examiner asserts that Klemm teach determining a time period for a thread (col. 5, lines 57-58 "thread execution time exceeds user-specified threshold"). Klemm seem to teach a user-specified threshold for a thread to run to avoid a thread hang. However, the threads Taught by Klemm are for a multi-threaded Java application and are not related to pre-fetching memory accesses, or helping a main thread. Thus, while the art may appear on its face to be analogous, the implementing threads according to Klemm will not result in the same type of result as the recited helper threads. Further, Klemm teaches terminating a thread when it has been determined to be hung (based on a pre-determined threshold), but Applicants terminate *helper* threads when it is predicted that the helper thread is not needed, in order to release resources that may be used by the main thread. Moreover, Claims 20-21 and 26-27 are dependent on allowable base claims, as discussed above.

### CONCLUSION

Applicants respectfully request reconsideration in view of the remarks and amendments set forth above. If the Examiner has any questions, the Examiner is encouraged to contact the undersigned at **703-633-6845**. Please charge any shortage of fees in connection with the filing of this paper, including extension of time fees, to Deposit Account 50-0221 and please credit any excess fees to such account.

Respectfully submitted,

Dated: April 30, 2010

/ Joni D. Stutman /  
Joni D. Stutman, Reg. No. 42,173  
Sr. Patent Attorney  
Intel Corporation  
703-633-6845

Intel Corporation  
c/o Intellevate, LLC  
P.O. Box 52050  
Minneapolis, MN 55402